COMAL Users Group, U.S.A., Limited
6041 Monona Drive
Madison, WI 53716

and

COMALites United
1670 Simpson #102
Madison, WI 53713

Monday, March 16, 1987

To: **COMAL Standards Group**

What is needed:

1) to have a user defined **ZONE** used between items in a **PRINT** statement.

2) a way to always print one space between items in a **PRINT** statement.

3) a way to always print no spaces between items in a **PRINT** statement.

Our proposal:

1) Keep the current **ZONE** statement (number 138 on page 28 of the April 1985 Kernal). This provides that the comma "," between items in a **PRINT** statement means to print spaces up to the next zone. The default zone is 0 (no spaces).

Example - set the zone to 5:

```
ZONE 5
```

or

```
z:=5
ZONE z
```

Example - change the zone to be 1 greater than it's current setting:

```
z:=ZONE
ZONE z+1
```

or

```
ZONE ZONE+1
```

Example - Print an addition chart:

```
ZONE 5
PRINT "add",1,2,3,4,5,6,7,8,9
FOR x:=1 TO 9 DO
  PRINT x;">",
  FOR y:=1 TO 9 DO
    PRINT x+y,
  NEXT y
  PRINT // gives carriage return at end of line
NEXT x
```

The result when running the above program:

```
add 1   2   3   4   5   6   7   8   9
1 > 2   3   4   5   6   7   8   9   10
2 > 3   4   5   6   7   8   9   10  11
3 > 4   5   6   7   8   9   10  11  12
4 > 5   6   7   8   9   10  11  12  13
5 > 6   7   8   9   10  11  12  13  14
6 > 7   8   9   10  11  12  13  14  15
7 > 8   9   10  11  12  13  14  15  16
8 > 9   10  11  12  13  14  15  16  17
9 > 10  11  12  13  14  15  16  17  18
```

2) Two items in a **PRINT** statement separated by a semi-colon ";" should always result in one space output between the two items. Note that it is a space and not just a cursor right movement. This is the same meaning as specified by the April 1985 Kernal (top of page 27 after number 130). It also matches the *COMAL Handbook* **PRINT** statement definition (page 223). The setting of the zone does not affect this.

Example: Print a statement:

```
DIM name$ OF 20
name$:="Sam"
PRINT "My name is";name$
```

The result of this program when run:

```
My name is Sam
```

If a new line were added to the top of this program:

```
ZONE 30
```

The result of the modified program would be the same as before.

3) Two items in a **PRINT** statement separated by an exclamation point "!" should always result in no space output between the items. The setting of the zone does not affect this.

Example: print a modular word:

```
DIM prefix$ OF 10, suffix$ OF 10, word$ OF 20
prefix$:="re"
suffix$:="ing"
word$:="build"
PRINT prefix$!word$!suffix$
```

The result of this program when run:

rebuilding

Our reasons:

It is good to keep the COMAL Kernal up-to-date. We do not want it to become stagnant. However, it should always strive to be upward compatible, whenever possible. We have heard that the meaning of the comma "," and the semi-colon ";" have been changed at the last Standards meeting. We are not sure, as we never received a copy of the new Kernal, nor the minutes to the meeting, and were not informed of the meeting until just a few weeks before it was to begin, and thus could not attend.

We understand the need to have **no space** between items in a **PRINT** statement regardless of the zone setting. But there is absolutely no justification to change the 1985 Kernal to accomplish it! There are many ways to accomplish it without causing many existing COMAL programs to malfunction. One possible method is outlined above.

We have a **very strong** feeling about this item. It is one of the reasons that we do not switch to the newest version of UniComal IBM PC COMAL and do not care for the UniComal C128 COMAL cartridge.

It is essential that the Kernal be returned to its original state before COMAL loses its credibility. We can in no way agree with the change made at the last meeting, and thus have submitted this proposal. Please enter our vote to approve this proposal or any modification to it that is **fully compatible** with the April 1985 Kernal, guaranteeing our programs already written will work with the updated Kernal.

Respectfully submitted by:

Len Lindsay, President
COMAL Users Group, U.S.A., Limited
Users Support Group for USA and Canada

David Stidolph, President
COMALites United, (Apple COMAL implementation)

Please excuse the short notice on this proposal, but we did not know about the April Standards meeting until today.

COMAL Users Group, U.S.A., Limited
6041 Monona Drive
Madison, WI 53716

and

COMALites United
1670 Simpson #102
Madison, WI 53713

Tuesday, March 17, 1987

To: **COMAL Standards Group**

What is needed:

1) a way to clear the screen, or current window if windowing is being used.

2) a way to issue a "*new page*" command to a printer or other device.

3) a way to have a program work properly to screen or printer without changing the program.

Our proposal:

Put the keyword **PAGE** into the Kernal. It is described on page 213 of the *COMAL Handbook*. It has no parameters. It's syntax is:

PAGE

If the current output location is the screen, the screen is cleared, and the cursor is placed at the first position of the top row on the screen. If windows are being used, only the current window is cleared, and the cursor placed in the first position of the top row of the current window. **PAGE** does not reset windows if they are used.

If the current output location is other than the screen, a **CHR$(12)** is issued to the output location. This is a page feed on most printers.

Our reasons:

Clearing the screen and paging a printer are two very common things that a program does. There should be a standard way to do this from COMAL. It should be a simple command, no parameters involved. UniComal, Mytech, and DalgaSoft follow this **PAGE** convention already, and Apple COMAL will do so as well.

COMAL Users Group, U.S.A., Limited
6041 Monona Drive
Madison, WI 53716

and

COMALites United
1670 Simpson #102
Madison, WI 53713

March 19, 1987

To: COMAL Standards Group

TeleNova's proposal #14 actually brings up several items, all related. We hope they all can be discussed as one unit. The four inter-related items are:

- ORD("")
- "" IN "abc"
- "" and CHR$(0)
- KEY$ *when no key is pressed*

We have several different implementations of COMAL here. We tried a few tests on them to see how compatible COMAL was between them. The results are summarized by this chart:

test	C64 - 0.14	C64 - 2.0	IBM PC	Unicomal PC	Mytech PC	CP/M
1» ORD("")	error	error	error	error	error	error
2» "" IN "abc"	4	4	4	0	1	4
3» "" = CHR$(0)	0	0	0	0	0	0
4» "" < CHR$(0)	1	1	1	1	1	1
5» CHR$(0) = ""	0	1	0	0	0	0
6» KEY$ *no key*	CHR$(0)	CHR$(0)	""	""	""	""

The results are interesting. First, all versions give an error for ORD(""). All versions also agree that "" <> CHR$(0) (nothing can't be equal to something) and that "" < CHR$(0) (nothing is less than something). However, it is interesting that UniComal C64 2.0 implemenation seems to contradict itself. "" = CHR$(0) is false, yet CHR$(0) = "" is true. More importantly, we are concerned that "" IN "abc" gives such a wide variety of results, and even not pressing a key during KEY$ can have different results.

First, we would like to look at what COMAL should return from KEY$ when no key is pressed. We believe that no key pressed is **nothing**, and that it is clear that the "" null string is nothing (the test #4 above shows that **all** implementations we tested agree that "" is nothing). Therefore, we hope all can agree that KEY$ should return "" when no key is pressed.

Now, look at test #2 in the chart above. Compatibility problems galore! Actually, we think that the TeleNova proposal #14 will be the key to solving all of this.

Having ORD("") return -1 is a rather good idea. Since it prevents a run-time error it could be quite helpful to COMAL programmers. Best yet, it has a very good side effect... it can help end the debate over "" IN "abc". And "" IN "abc" is a very, very fundamental issue. It is the basis for getting a reply from the user without requiring that they also press the «*return*» key as well (via INPUT). For example:

```
PRINT "Shall we reformat the hard disk?"
REPEAT
   reply$:=KEY$
UNTIL reply$ IN "yYnN"
IF reply$ IN "yY" THEN format'hard'disk
```

Now, if "" IN "abc" is TRUE, (and no key pressed returns "") then... If the user does not press a key within a microsecond of the prompt, the REPEAT loop will be executed only once, since the UNTIL condition was met. It then finds the IF condition TRUE as well, and reformats your hard disk ... and you didn't even hit a key yet!!!

Now, the TeleNova proposal solves this dilema!

Let's look at how we might scan a string for a match to a one character reply:

Let's say that reply$ is equal to "c", and that valid$ is equal to "abc".

First, we note that ORD("c") is 67.

Now, we simply look at each character in valid$, one at a time, until we find one that has the ORD value of 67. We find our match at the third character, so we return a value of 3.

Now, let's try that again under TeleNova's proposal for ORD("") = -1... but this time let's have reply$ be ""... this should be interesting...

First, we note that ORD("") is -1.

Now, we simply look at each character in valid$, one at a time, until we find one that has the ORD value of -1. However, none of the characters in valid$ have an ORD value of -1, so we return a value of 0 (meaning false).

So, if ORD("")=-1 then "" IN "abc" is false.

YAY! That is what the users want. An ordinary person (that is who COMAL was designed for) would expect that **no reply** would not match a list of valid reply's. How can nothing match something **exactly**. After all, IN requires an **exact** match.

With that in mind, looking at the chart above reveals that the latest UniComal IBM PC COMAL provides a false response to "" IN "abc".

We propose that it be prominently noted in the Kernal that:

1) "" IN "any string length greater than 0" return a value of 0 (false)
2) Not pressing any key results in KEY$ returning the "" null string.
3) "" is **not** equal to CHR$(0).

Respectfully submitted for your approval.